Bevezetés a lágy számítás módszereibe

Genetikus algoritmusok
Rekombináció, mutáció
.M fájlok készítése a MATLAB-ban

Werner Ágnes
Villamosmérnöki és Információs Rendszerek Tanszék

Rekombináció

- szelekció után keletkező szülő állomány
  - tartalmazhat ismétlődéseket
  - részben vagy egészben azonos lehet a populációval
- kettő-több szülő felhasználásával képez utódot (jellemzően kettőből egyet vagy kettőt)
- célja: a szülőkből minél jobb, újabb megoldások összeállítása az átvett, "örökölt" tulajdonságok alapján
- formáját befolyásolja a változók típusa és a problémák sajátosságai
- $P_r \approx 0,7$

Diszkrét rekombináció

- többféle változó típus esetén alkalmazható: egész, valós, bináris, sztring, szimbólumok
- a művelet csak néhány diszkrét értékkel dolgozhat
- két szülő változóiból véletlenszerűen képezzük az utódot
- jelölje $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$ és $(u_1, u_2, \ldots, u_n)$ a két szülő és az utód egyedet változóikkal együtt
- minden változónál képezzük az

  $u_i = ax_i + (1-a)y_i \quad i = 1, 2, \ldots, n$ műveletet, ahol $a \in \{0, 1\}$ véletlen szám és minden változó esetén újra generálásra kerül
- bővített változat

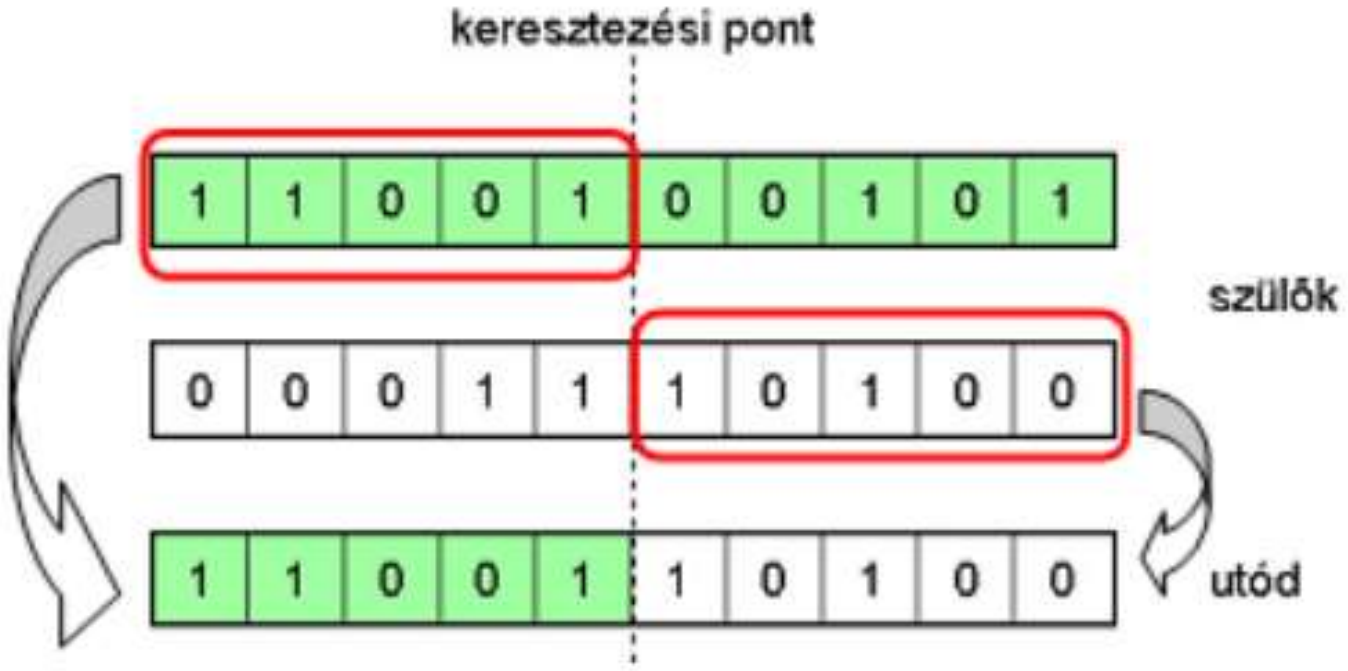
Egész és valós típusú változók rekombinációja

- **Köztes rekombináció**
  - $u_i = ax_i + (1-a)y_i \quad i = 1, 2, \ldots, n)$, ahol $a \in [-h, 1+h]$ véletlen szám és a minden változó esetén újra generálásra kerül a h értékét általában $0,5$-nek választjuk
  - bővített változat
- **Lineáris rekombináció**
  - $u_i = ax_i + (1-a)y_i \quad i = 1, 2, \ldots, n)$, ahol $a \in [-h, 1+h]$ véletlen szám és minden változó esetén ugyanazon a értéket alkalmazzuk
  - speciális változat $a = 0,5$

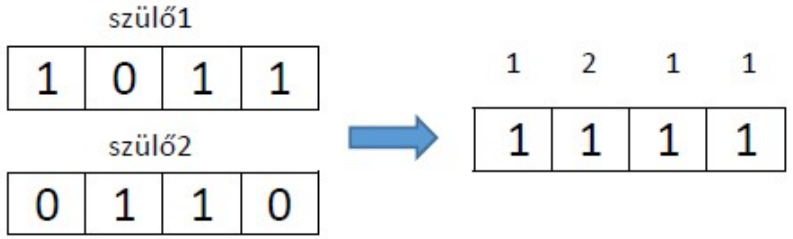
Bináris sztringek rekombinációja

- **Egypontos keresztezés**
  - két szülőből két utód
  - véletlenszerűen választunk keresztezési pontot az $\{1, 2, \ldots, L-1\}$ pozíciók közül
- **Többpontos keresztezés**
  - két szülőből két utód
  - n számú keresztezési pontot választunk
  - a kapott keresztezési pontokat növekvő sorrendbe rendezzük, majd a megfelelő, egymás után következő keresztezési pontok közti bitsorozatokat rendre más-más szülőtől választjuk

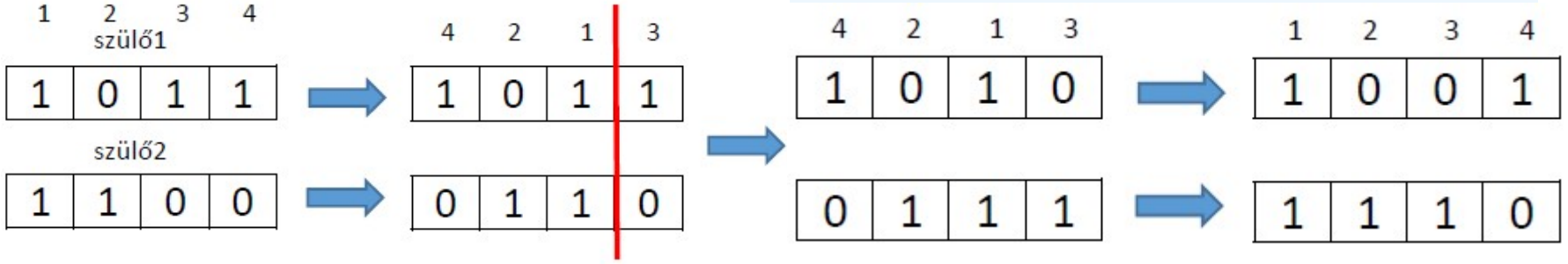
Egypontos keresztezés



Bináris sztringek rekombinációja



- **Uniform keresztezés**
  - minden bit pozíción külön-külön döntjük el, melyik szülőtől választjuk a következő bit értékét
- **Keverő keresztezés**
  - mindkét szülőben a bit pozíciókat azonos módon, véletlenszerűen összekeverjük
  - egypontos keresztezést alkalmazunk
  - visszaállítjuk a bitpozíciók eredeti sorrendjét



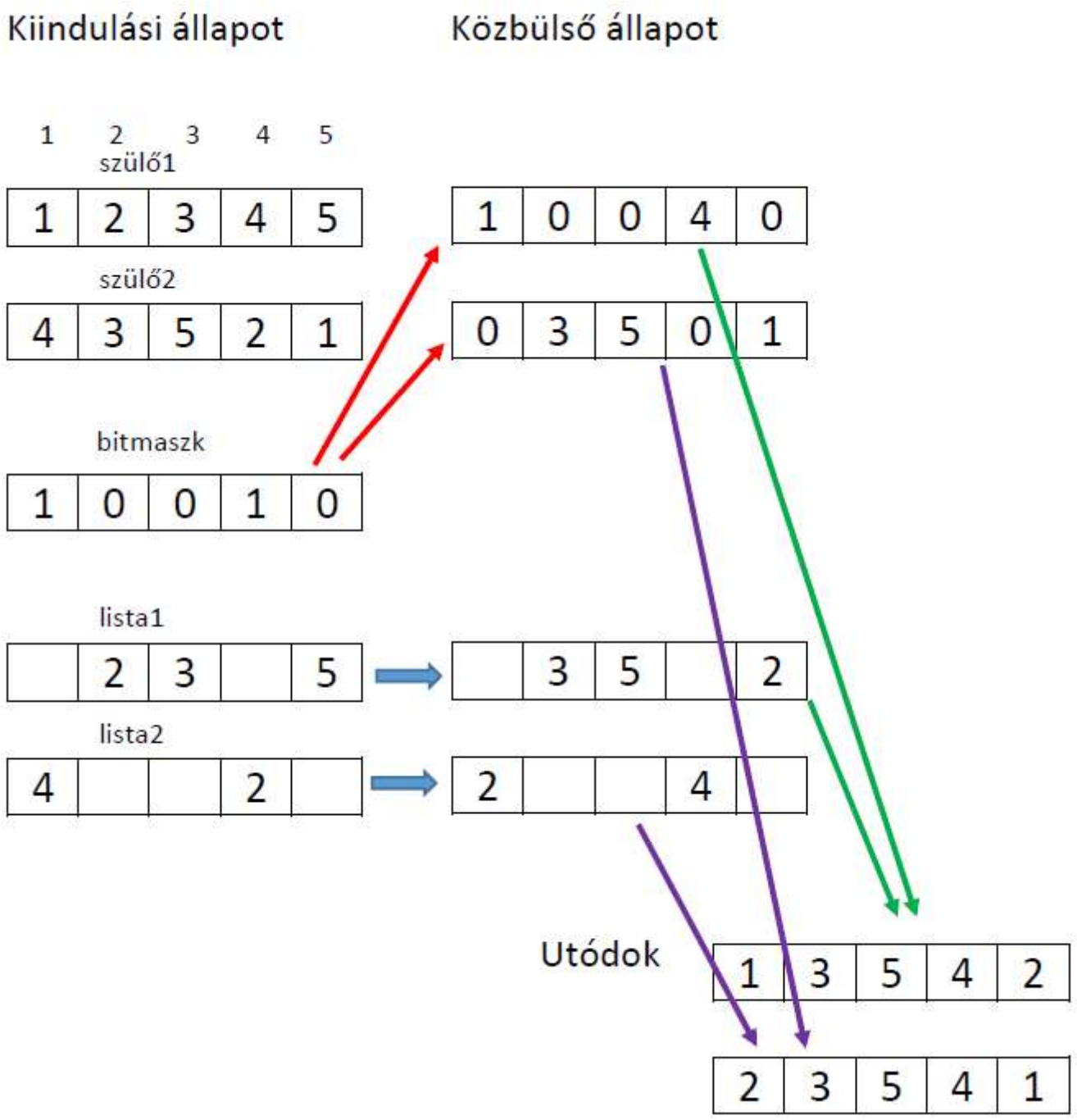
Permutációk rekombinációja

az egyedek csak érvényes permutációk lehetnek

Uniform sorrend alapú rekombináció

2 szülőből 2 utódot állít elő és a szülők relatív sorrendjét örökíti át

1. előállít egy bit maszkot, amely minden pozíción véletlenszerűen 0 vagy 1 értéket tárol; azon pozíciókon, ahol 1 értéket tárol a bitmaszk, az 1. szülő megfelelő pozícióján lévő permutáció sorszámát átmásolja az 1. utód azonos pozíciójára
2. az 1. utód többi pozícióját üresen hagyja
3. a 2. utódot hasonlóan állítja elő, csak itt a 0 bitmaszk értékű pozíciókat veszi elő és a 2. szülő megfelelő pozícióján lévő permutáció sorszámát másolja át a 2. utód azonos pozícióira
4. a 2. utód többi pozícióját üresen hagyja (az eredmény egy köztes állapot, amelyben mindkét utódnál csak részben ismerjük a keresett permutációt)
5. az 1., majd a 2. utódban tölti fel a hiányzó helyeket, ehhez listát készít az 1. szülő azon sorszámairól, amelyek nem kerültek át az 1. utódba
6. a listát úgy rendezi, hogy minél több sorszámegyezést érjen el a 2. szülő azonos pozícióin
7. a kész lista elemeivel balról-jobbra haladva feltölti az 1. utód üres pozícióit, a 2. utód üres pozícióit hasonló módon tölti fel


Kiindulási állapot
Közbülső állapot
1 2 3 4 5
szülő1
1 2 3 4 5
szülő2
4 3 5 2 1
bitmaszk
1 0 0 1 0
1 0 0 4 0
0 3 5 0 1
lista1
2 3 5
3 5 2
lista2
4 2
2 4
Utódok
1 3 5 4 2
2 3 5 4 1

Mutáció

A rekombináció nem alkalmas finom közelítések megvalósítására.
A mutáció az utód közvetlen környezetében keres jobb megoldásokat.

$P_m \approx 0,01$

Valós és egész típusú változók mutációja

különbség a mutációs lépés nagyságában
Schlierkamp-Voosen és Mühlenbein mutáció művelete:

az x_i változóból a $z_i = x_i \pm range_i * \delta$

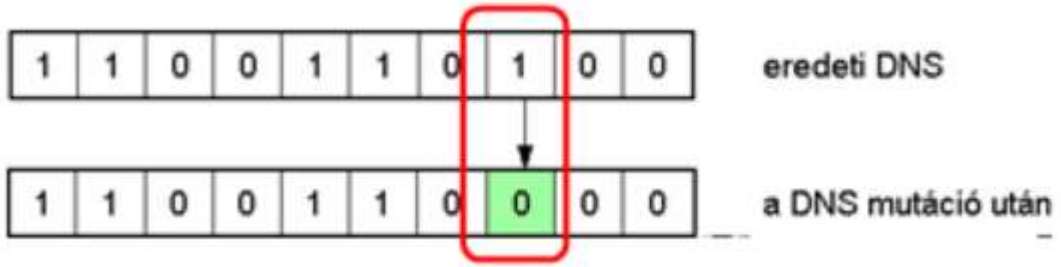
a $+$ vagy $-$ előjelet $0,5$ valószínűséggel választjuk,
a $range_i$ az x_i változó szomszédsági környezetének szélességét jelöli,
a δ a pontosságot határozza meg, diszkrét vagy folytonos értékei lehetnek

Bináris típusú változók mutációja

az egyed egy bitsorozat, melynek egyes bitjeit mutációval változtatjuk

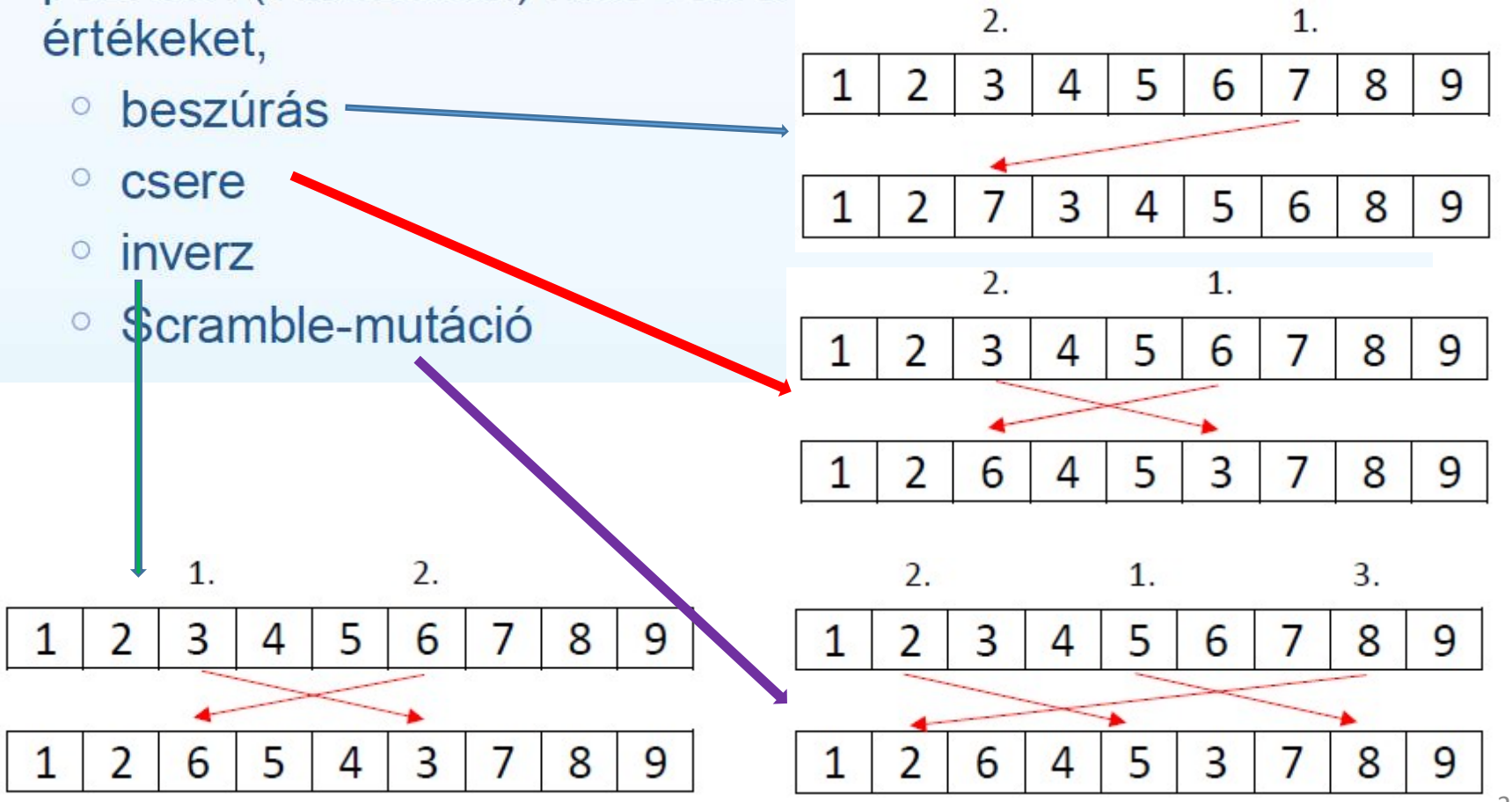
a mutáció egy x_i változónál a következő:

$$z_i = \begin{cases} x_i & , ha \;\; Rnd > P_m \\ 1 - x_i & , ha \;\; Rnd \leq P_m \end{cases}$$



Permutációk mutációja

- jelöljék az egyed változói a permutáció egyes pozícióin található értékeket,
- a mutáció ezt a sorrendet úgy változtatja meg, hogy néhány pozíción (változóban) más sorrendben helyezi el az értékeket,
  - beszúrás
  - csere
  - inverz
  - Scramble-mutáció



Visszahelyezés

A szelekció, rekombináció, mutáció műveletsorral minden generációban új utódokat kapunk.
Ezen utódokkal, vagy egy részükkel bővíteni kell a populációt, lecserélve velük a korábbi egyedek egy részét.
bevezetünk 2 rátát:

- utódképzési ráta
- visszahelyezési ráta
- relációk a ráták között:
  - $utodkepzesirata = visszahelyezesirata = 1$
  - $utodkepzesirata \leq visszahelyezesirata < 1$
  - $utodkepzesirata > visszahelyezesirata$

Elitizmus

A ciklus kialakítása

- **stratégiai paraméterek megadása**
  - populáció mérete
  - a rekombináció alkalmazásának P_r valószínűsége
  - a mutáció alkalmazásának P_m valószínűsége
  - az utódképzési ráta értéke
  - a visszahelyezési ráta értéke
- **kezdő populáció kialakítása**
  - véletlenszerű előállítás
  - előző feladat eredményeinek felhasználása
  - korábbi eredmények módosított felhasználása

Megállási feltételek

- maximális generációszám elérése
- maximális futási idő elérése
- adott idő alatt nem javul a megoldás minősége
- hasonlóak az egyedek
- előre adott érték megközelítése
- a populáció minősége megfelelő (mérőszámok):
  - a célfüggvény értékének standard szórása az aktuális generációban
  - a célfüggvény értékek átlagának és a legjobb értéknek az eltérése az aktuális generációban
  - a legjobb és a legrosszabb célfüggvény érték eltérése az aktuális generációban

Fitnesz kiértékelés

A problémák legtöbbjénél ismerünk egy **célfüggvényt**, amely az értelmezési tartomány, azaz a keresési tér pontjaihoz egy valós számot vagy vektort rendel.

Az esetek többségében a fitneszfüggvényt azonosnak választjuk a célfüggvénnyel.

Sokszor nincs célfüggvényünk és a fitneszfüggvény megfogalmazása a probléma megoldásának egyik fontos kulcseleme.

Konkrét formája függ a reprezentációtól.

Költség és hatékonyság elemzés

- Nehéz egy olyan módszer hatékonyságát elemezni, ami nemdeterminisztikus lépéseket használ működése közben.
- Egy adott GA implementáció futása a véletlen elemek miatt mindig változik, nincs két egyforma kimenetű futás (hacsak nem nagyon triviális az algoritmus, vagy nagyon rossz a véletlenszám-generátor adott megvalósításnál).
- Így nem sok értelme lenne összehasonlító futási eredményeket nézegetni, főleg olyan példák esetében ahol az abszolút futási idő legdurvább esetben is csak néhány másodperc.
- Nézzük meg milyen tényezők befolyásolják a GA költségét!

Populáció mérete

A populáció mérete nyilvánvalóan alapvetően befolyásolja mind a futás, mind a tárhely igényét.

Generációk száma

A generációk számával egyenesen arányos a futási idő, mivel minden új generáció egy új főciklus lefuttatását igényli. Mivel a kilépési feltétel nemcsak a maximális generációszámtól függ, előbb is terminálhat az algoritmus, nem feltétlenül igaz, hogy egy *10x* nagyobb generációszámmal paraméterezett algoritmus *10x* annyi ideig fut.

Szelekció

A szelekció GA esetében a populáció méretének függvénye, mivel pont ennyi szülőt kell kiválasztanunk (elitista módszer esetén ennek egy részét). Ez a megvalósításoktól függően általában lineáris költségű, de bizonyos esetekben (pl. pár-verseny szelekció) lehet nagyobb is.

Rekombináció, mutáció

Hasonlóan a szelekcióhoz, ez is a populáció méretének a függvénye. Viszont itt már bizonyos esetekben a futási költségbe beleszólhat az egyedek reprezentációja. Ugyanis több esetben a gének számával arányos az egyedek keresztezése ill. mutációja. Természetesen több gén esetén tovább tart ezek elvégzése.

Visszahelyezés

A visszahelyezés művelete $\sigma(1)$ általában, mivel egyszerűen csak a régi populációt megfeleltetjük az újonnan kialakítottal. Olyan esetekben viszont, amikor az algoritmus nem generációs, lehetséges nagyobb költségű megvalósítás is.

Fitnesz kiértékelés

A fitnesz kiértékelés nagyban függ a kiértékelő függvénytől. Vizsgált eseteinkben ez $\sigma(1)$, de könnyen elképzelhető igen bonyolult fitnesz függvény is.

Nézzünk meg egy nagyon egyszerű összehasonlítást

Globális minimum

keresése az első példánkban vett $f(x) = x^2$ függvénynél.

Versenyeztessük meg a **ga** és az **fminsearch** algoritmusokat!

A feladat komplexitása miatt azt várjuk, hogy az utóbbi lesz a nyertes.

Lefuttatva alapbeállításokkal, 100 futás átlagára azt kapjuk, hogy kb. 12-13-szor gyorsabb, mint a GA megvalósítás.

Ráadásul az eredmény átlag 15-16 jegyre pontos, míg a genetikus metódus 4-7 jegyig egyező eredményeket tud csak produkálni.

Javítva a GA paraméterezésén, némi kísérletezés után ezt sikerül annyira lefaragni, hogy azonos pontosság mellett mindössze 5-ször annyi időbe telik lefuttatni a ga parancsot, mint társát.

Mit mutat ez meg?

- Nem többet és nem is kevesebbet, mint amire számítottunk.
- Látszik, hogy ilyen típusú feladatnál érdemesebb a hagyományos kereső technikákat alkalmazni, főleg hogy ezeknek igen jó megvalósítása adva van Matlab környezetben. A probléma túl egyszerű, túl költséges egy párhuzamos számításokat alkalmazó eljárás használatához.
- Érdemes viszont azt megfigyelni, hogy a paraméterek helyes beállítása milyen nagymértékben tudja befolyásolni a futás költségét és kimenetelét.

További következtetések

- További futási idő teszteken azt kapjuk, hogy Rastriginfüggvény esetében megmarad az fminsearch 10-15-szörös sebesség különbsége, alapbeállítások mellett. Cserébe nem ad helyes eredményt egyikre sem.
- Ebből olyan következtetést lehet levonni, hogy a GA alkalmazása olyan esetekben célszerű ahol jobb eredményt várunk tőle, mint a hagyományos kereső algoritmusoktól, vagy ahol azok nem használhatóak.
- Bizonyos esetekben jó megoldás lehet az algoritmusok keverése: egy alacsonyabb költségű kereső eljárás segítségével információt szerzünk a problémáról, amit felhasználhatunk a későbbi GA paraméterezésénél.